Python 数据分析与数据挖掘（Python for Data Analysis&Data Mining）

Chap 5 网络数据收集和分析 Web Data Collection and Analysis

内容：

- 数据分析实战：本地数据和Web数据

实践：

- Web数据获取
- 股票金融数据分析
- 可视化绘图
- 不同文件格式的磁盘文件读取和保存
- 文件格式（csv，excel，json，html等）

上节课讲述了从本地和Web获取数据，并进行数据的预处理、分析、可视化和磁盘保存。

本章目录：

- 1) 读入本地磁盘数据，并进行数据分析，绘图
- 2) 获取Web数据（股票数据），并进行股票数据的分析和处理，保存磁盘

In [ ]:
```
# 必要准备工作：导入库，配置环境等
#from __future__ import division
#import os, sys

# 导入库并为库起个别名
import numpy as np
import pandas as pd
from pandas import Series, DataFrame

# 启动绘图
%matplotlib inline
import matplotlib.pyplot as plt
```

本地数据集1：餐馆小费

tips.csv 是个关于餐馆小费记录的数据，包含七个字段
（total_bill，tip，sex，smoker，day，time，size），共计244条记录。

- 磁盘读入csv格式文件转为pd数据结构
- 对数据分析（缺失值-填充，清理，汇总描述，可视化绘图）
- 数据相关性分析
- 数据分组聚合

In [ ]:
```
import pandas as pd
tips = 'data/tips.csv'
data = pd.read_csv(tips)  #默认header='infer'，推导第一行是header，小费记录始于第二行
print len(data) #数据记录数量
data.head()  #预览最前5行记录
#data.tail() #预览最后5行记录
```

In [ ]:
```
data.describe() #数据的汇总描述
```

In [ ]:
```
stat = data.describe() #数据的基本统计量
#重点：可以自己添加统计量信息
stat.loc['range'] = stat.loc['max'] - stat.loc['min']  #极差 range
stat.loc['var'] = stat.loc['std'] / stat.loc['mean']  #变异系数，标准差/均值的离中趋势
stat.loc['dis'] =  stat.loc['75%'] - stat.loc['25%'] #四分位数间距（极差）
stat
```

In [ ]:
```
#数据的其他统计量
#数据的分布统计
data.median()  #数据的中位数
data.mode() #数据的众数
data.quantile(0.1)  #数据的百分位数 data.quantile(q=0.5)

#数据的离中趋势度量
data.skew() #数据的偏度
data.kurt() #数据的峰度

#数据列之间的相关度量
data.cov() #数据的协方差矩阵
data.corr() #数据的Pearson相关系数矩阵
```

In [ ]:
```
data.columns
```

In [ ]:
```
data.columns.names  #此时每个列没有name
```

In [ ]:
```
#启动绘图
%matplotlib inline
import matplotlib.pyplot as plt

data.boxplot(return_type='axes')  #画盒图，直接使用DataFrame的方法
#data.boxplot() #画盒图，直接使用DataFrame的方法，需要屏蔽warning
#data[['tip','size']].boxplot(return_type='axes') #只对两个列画盒图
```

In [ ]:
```
data.plot.line()  #线图
#data.plot.hist() #直方图
```

In [ ]:
```
#几种盒图绘图
#data[['tip','size']].boxplot() #盒图1，同前面
#data[['tip','size']].plot(kind='box') #盒图2
data[['tip','size']].plot.box()  #盒图3
```

In [ ]:
```
#其他多种图类型, tip, size, total_bill
#data['tip'].plot.line() #线图 1
```

```
#data['tip'].plot(kind='line') # 线图 2
#data['tip'].plot.hist() # 直方图
#data['tip'].plot.kde() # 密度图1 'kde' : Kernel Density Estimation plot
#data['tip'].plot.density() # 密度图1 density,同上
#data['tip'].plot.pie() # 饼图
```

In [ ]:
```
# 相关性分析
data.corr()  # method : {'pearson', 'kendall', 'spearman'}，默认pearson
```

In [ ]:
```
data.corr(method='kendall')  # method : {'pearson', 'kendall', 'spearman'}
```

In [ ]:
```
data.head()
```

思考： 数据分组进一步考虑：消费与date（周一周末）是否有关？是否与time（中餐晚餐）有关？

怎么做？

- 把day和time两个列转为行索引的外层和内层
- DataFrame的set_index函数会将其一个或多个列转换为行索引，并创建一个新的DataFrame。

In [ ]:
```
# 考虑bill与date（周一周末）是否有关？是否与time（中餐晚餐）有关？
# 把day和time两个列转为行索引的外层和内层
# DataFrame的set_index函数会将其一个或多个列转换为行索引，并创建一个新的DataFrame。
data2 = data.set_index(['day', 'time'])
data2.head()
```

In [ ]:
```
# 选取周日Sun的消费统计汇总情况
data2.ix['Sun'].describe()
```

In [ ]:
```
# 选取周日Sun晚餐Dinner的消费统计汇总情况
data2.ix['Sun'].ix['Dinner'].describe()
#data2.ix['Sun'].ix['Lunch'].describe() # 周日没有Lunch消费的记录
```

In [ ]:
```
# 对比工作日周五的午餐和晚餐消费均值
print data2.ix['Fri'].ix['Lunch'].mean() # 选取周五Fri午餐Lunch的消费统计汇总情况
print data2.ix['Fri'].ix['Dinner'].mean() # 选取周五Fri晚餐Dinner的消费统计汇总情况
```

In [ ]:
```
# 对比周五到周日的消费均值
print data2.ix['Fri']['total_bill'].mean()
print data2.ix['Sat']['total_bill'].mean()
print data2.ix['Sun']['total_bill'].mean()
```

In [ ]:
```
# 交换索引（行索引的内层和外层索引交换）
data3 = data2.swaplevel(0,1)  # 交换索引后返回新的data3
data3.tail()
```

In [ ]:
```
# 比较午餐Lunch和晚餐Dinner的消费统计汇总情况
print data3.ix['Dinner'].describe()
print data3.ix['Lunch'].describe()
```

其实，我们不必进行上面的操作，因为pandas提供了非常方便的groupby分组操作

groupby分组操作

pandas的DataFrame有groupby操作，可以非常方便对数据分组。不需要将多个列索引转换为行索引的情况下，可以直接对数据进行分组分析计算。

- df.groupby(['col2', 'col3']) # 首先，按照col2和col3的不同值进行分组
- df['col1'].describe() # 然后，统计col1的汇总情况

In [ ]:
```
# 添加“小费占总额百分比”列
data['tip_pct'] = data['tip'] / data['total_bill']
data[:6]
```

In [ ]:
```
# 分组统计
data.groupby(['sex','smoker']).count() # 统计不同性别和是否抽烟的数量
```

In [ ]:
```
# 分组统计
data.groupby(['day','time']).count()  # 统计不同天和不同餐时的数量
```

In [ ]:
```
# 统计不同天和时间的平均情况
data.groupby(['day','time']).mean()
```

In [ ]:
```
# 只统计不同天和时间的tip平均情况
data['tip'].groupby([data['day'], data['time']]).mean()
```

In [ ]:
```
# 比较不同性别在不同天的午餐Lunch和晚餐Dinner的平均小费情况
data['tip'].groupby([data['sex'], data['time']]).mean()
```

In [ ]:
```
# 综合比较不同性别在周末午餐Lunch和晚餐Dinner的平均消费情况
data.groupby(['sex', 'time']).mean()
```

In [ ]:
```
# 分组统计不同性别给出小费的比例情况
grouped = data.groupby(['sex'])
grouped.mean()
```

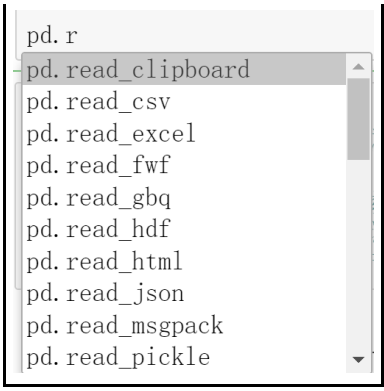
In [ ]:
```
# 不同性别给小费比例的均值
grouped['tip_pct'].mean()
```

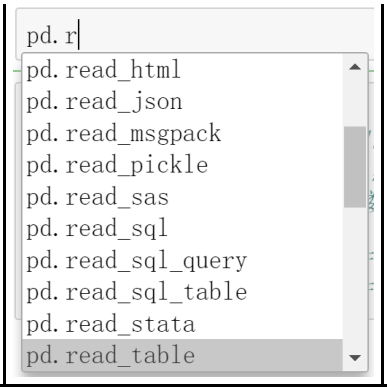
In [ ]:
```
# 首先，根据sex和smoker对tips进行分组
grouped = data['tip_pct'].groupby([data['sex'],data['smoker']])
grouped.mean()
```

小作业1-a

- 考虑并分析其他特征或特征组合，如性别、是否抽烟、就餐人数等与消费量和消费习惯等的情况
- 对各种分析的可视化展示

pandas可以读入的数据文件格式包括：





pandas读入多种数据格式：
- **csv, excel**
- **html, json**
- **pickle**
- 剪贴板
- 数据库**, sql, hdf**
- **SASXport**，**Google BigQuery**等
- 通用表格**, table**

本地数据集**2**：股票时间序列数据

2 - stock_px.csv 是个关于股票股价记录的时间序列数据，包含9个字段（AA，AAPL，GE，IBM，JNJ，MSFT，PEP，SPX，XOM），时间从1990/2/1到2011/10/14，共计5472条记录。

美铝公司[AA]，苹果公司[AAPL]，通用电气[GE]，微软[MSFT]，强生[JNJ]，百事[PEP]，美国标准普尔500指数 (SPX)，埃克森美孚[XOM]

In [ ]:
```
import pandas as pd
import numpy as np
from datetime import datetime

f = 'data/stock_px.csv'
data = pd.read_csv(f, index_col='date')  # 使用date列作为行索引
data.index = pd.to_datetime(data.index)  # 将字符串索引转换成时间索引
print len(data) # 数据记录数量
data.head()  # 预览最前5行记录
data.tail()  # 预览最后5行记录
```

In [ ]:
```
plt.rc('figure', figsize=(12,6))
data[['AAPL','IBM','MSFT','GE','AA','JNJ','PEP']].plot.line() #绘曲线图
```

In [ ]:
```
# 重点了解苹果股票的基本统计量
data['AAPL'].describe()
```

In [ ]:
```
# 苹果股票的其他统计量
print data['AAPL'].median()  # 数据的中位数
print data['AAPL'].mode() # 数据的众数
print data['AAPL'].quantile(0.1)  # 数据的百分位数 data.quantile(q=0.5)
print data['AAPL'].skew() # 数据的偏度
print data['AAPL'].kurt() # 数据的峰度
```

In [ ]:
```
# 各股票之间的相关统计量
#data.cov() # 数据的协方差矩阵
data.corr() # 数据的Pearson相关系数矩阵
```

In [ ]:
```
data.head()
```

In [ ]:
```
data.tail()
```

股票的时间序列数据分析

除了前面常用的统计、汇总、分组、可视化分析，对于股票数据，还可以进行更复杂的数据分析任务：

- 根据每天的收盘价返回对数收益率

In [ ]:
```
# 只取出苹果股票分析
df = pd.DataFrame(data['AAPL'],index=data.index, columns=['AAPL'])  # data['AAPL']只是个Series
df.tail()
```

In [ ]:
```
# 更复杂的数据分析任务：根据每天的收盘价返回对数收益率
# 首先添加包含对应信息的列，生成一个新的列，
# 然后中所有股价上进行循环，逐步计算单个对数收益率值
df['Return'] = 0.0
for i in range(1, len(df)):
    df['Return'][i] = np.log(df['AAPL'][i] / df['AAPL'][i-1])
df.tail()
```

In [ ]:
```
# 也可以使用向量化代码，在不使用循环的情况下得到相同的结果，即shift方法
df['Return2'] = np.log(df['AAPL'] / df['AAPL'].shift(1))
df[['AAPL', 'Return', 'Return2']].tail()
# 最后面两列的值相同：更紧凑和更容易理解的代码，而且是更快速的替代方案
```

In [ ]:
```
# 目前，一个对数收益率数据列就足够了，可以删除另一个列
del df['Return2'] # 删除列
df.tail()
```

In [ ]:
```
# 绘图更好地概览股价和波动率变化
df[['AAPL', 'Return']].plot(subplots=True, style=['b','r'], figsize=(8,5))
```

In [ ]:
```
# 技术型股票交易者可能对移动平均值（即趋势）更感兴趣，
# 移动平均值很容易使用pandas的rolling_mean计算
df['42d'] = pd.rolling_mean(df['AAPL'], window = 42)
df['252d'] = pd.rolling_mean(df['AAPL'], window = 252)
df[['AAPL', '42d', '252d']].tail()
```

In [ ]:
```
df[['AAPL', '42d', '252d']].head() # 对于后两列，前面的数据为空
```

In [ ]:
```
# 包含两种趋势的典型股价图表绘图
df[['AAPL', '42d', '252d']].plot(figsize=(8,5))
```

In [ ]:
```
# 期权交易者更喜欢的话题，对数收益率的移动历史标准差--即移动历史波动率
import math
df['Mov_Vol'] = pd.rolling_mean(df['Return'], window=252) * math.sqrt(252)
df['Mov_Vol'].tail() # Mov_Vol列的数据在前面252行为空
```

In [ ]:
```
# 杠杆效应假设，说明市场下跌时历史移动波动率倾向于升高，而在市场上涨时波动率下降
df[['AAPL', 'Mov_Vol', 'Return']].plot(subplots=True, style=['b','g','r'], figsize=(8,8))
```

In [ ]:
```
# 包含两种趋势的典型股价图表绘图
```

```
df[['AAPL', '42d', '252d']].plot(figsize=(8,5))
```

In [ ]:
```
df.tail()
```

保存数据

现在我们想把对苹果股票的分析数据保存下来，在以后的分析中继续使用。pandas的DataFrame的保存数据类型，参考前面表格中的读入数据类型。

In [ ]:
```
# 为了以后更容易导入数据，我们生成一个新的csv数据文本，并将所有数据行写入新文件
out_file = open('data/aapl.csv', 'w')
df.to_csv(out_file)
out_file.close()
```

In [ ]:
```
# 使用Yahoo Finance的API获取四个公司的股票数据
import pandas as pd
import numpy as np
from pandas_datareader import data

codes = ['AAPL', 'IBM', 'MSFT', 'GOOG']  # 四个股票
all_stock = {}
for ticker in codes:
    all_stock[ticker] = data.get_data_yahoo(ticker)# 默认从2010年1月起始, start
='7/1/2005

volume = pd.DataFrame({tic: data['Volume'] for tic, data in all_stock.it
eritems()})
open = pd.DataFrame({tic: data['Open'] for tic, data in all_stock.iterit
ems()})
high = pd.DataFrame({tic: data['High'] for tic, data in all_stock.iterit
ems()})
low =  pd.DataFrame({tic: data['Low'] for tic, data in all_stock.iterite
ms()})
close =  pd.DataFrame({tic: data['Close'] for tic, data in all_stock.ite
ritems()})
price = pd.DataFrame({tic: data['Adj Close'] for tic, data in all_stock.
iteritems()})
```

In [ ]:
```
all_stock['AAPL'].head()
```

In [ ]:
```
price.head()
```

In [ ]:
```
all_stock['AAPL'].head()
```

In [ ]:
```
AAPL = all_stock['AAPL']
len(AAPL)
AAPL.head()
```

In [ ]:
```
# 为了以后更容易导入数据，我们生成一个新的csv数据文本，并将所有数据行写入新文件
AAPL.to_csv('data/AAPL-0.csv')
```

In [ ]:
```
# 有时网络访问不好，因此读入已经保存的AAPL股票数据
import numpy as np
import pandas as pd
f = 'data/AAPL-0.csv'
```

```
data = pd.read_csv(f, index_col='Date')  #使用date列作为行索引
data.index = pd.to_datetime(data.index)  #将字符串索引转换成时间索引
AAPL = data
print len(AAPL)
AAPL.tail()
```

In [ ]:
```
#包含两种趋势的典型股价图表绘图
AAPL[['Open']].plot(figsize=(8,5))
```

作业**1-b**:

- 从Yahoo！Finance下载美交所中国各种题材股票（阿里，百度，京东等等）
- 从Yahoo！Finance下载沪深交所各种题材股票（沪市SS，深市SZ）
- 分析并观察各种题材股票(保险类，新能源类，互联网相关)的各种统计情况、趋势、相关性分析等

In [ ]:
```
# 使用Yahoo Finance的API获取沪深股市的股票数据
import pandas as pd
import numpy as np
from pandas_datareader import data

#获取
Maotai = data.get_data_yahoo('600519.SS') #茅台股票代码+沪市
Maotai.tail()
```

In [ ]:
```
# 使用Yahoo Finance的API获取沪深股市的股票数据
import pandas as pd
import numpy as np
from pandas_datareader import data

#获取
PUFA = data.get_data_yahoo('600000.SS') #浦发银行股票代码600000+沪市SS
PUFA.tail()
```

In [ ]:
```
# 使用Yahoo Finance的API获取沪深股市的股票数据
import pandas as pd
import numpy as np
from pandas_datareader import data

#获取探路者股票代码是：300005，深市SZ
EXP = data.get_data_yahoo('300005.SZ') #探路者股票代码是：300005，创业板股票代码以300打头
EXP.tail()
```